Tackling the PAN'09 External Plagiarism Detection Corpus with a Desktop Plagiarism Detector[*]

James A Malcolm
University of Hertfordshire
College Lane, Hatfield, Herts
j.a.malcolm@herts.ac.uk

Peter C R Lane
University of Hertfordshire
College Lane, Hatfield, Herts
p.c.lane@herts.ac.uk

Abstract: Ferret is a fast and effective tool for detecting similarities in a group of files. Applying it to the PAN'09 corpus required modifications to meet the requirements of the competition, mainly to deal with the very large number of files, the large size of some of them, and to automate some of the decisions that would normally be made by a human operator. Ferret was able to detect numerous files in the development corpus that contain substantial similarities not marked as plagiarism, but it also identified quite a lot of pairs where random similarities masked actual plagiarism. An improved metric is therefore indicated if the "plagiarised" or "not plagiarised" decision is to be automated.

Keywords: Ferret, trigrams, plagiarism detection

1 *Introduction*

In this paper we describe how we approached the challenge of the PAN'09 Plagiarism Detection Competition using the Ferret plagiarism detection software. We outline Ferret's strengths in normal use, highlight the difficulties we had in using Ferret for the competition task, and describe the results of the improvements that we made as a result of entering the competition.

The "external plagiarism analysis" task of the PAN'09 Plagiarism Detection Competition (International Competition on Plagiarism Detection, 2009) is an example of category 1 plagiarism (Lyon and Malcolm, 2002), as we have the source(s) in our hand. This suggests that Ferret is the tool for the job.

Ferret (Lyon, Barrett, and Malcolm, 2004; Lyon, Malcolm, and Barrett, 2005) is a tool that (when used on student work) is primarily good for detecting collusion rather than plagiarism (though it has been extended to generate search terms to drive an Internet search (Malcolm and Lane, 2008b)). It is a desktop plagiarism detector, which means that it is fast and interactive. It has to be fast, because a human is waiting for the results, and because it is interactive, human input is available and appropriate: after all, plagiarism is an academic judgement not something that can be measured by a machine (Flint, Clegg, and Macdonald, 2006; Lyon, Barrett, and Malcolm, 2004). Before tackling the competition we had done little to automate the decision making process: "is this plagiarism or not"; Ferret tells its user *which* pairs to look at (and helps in reviewing those pairs) but leaves the actual decision has to him or her. There is therefore no need for the software to draw a dividing line – a ranked list is sufficient.

The competition did highlight what we knew to be Ferret's strengths and weaknesses: we came second on recall, but precision was poor as Ferret is *too* fine grained in its identification of similarities.

2 *Ferret's Strengths*

Assuming, as in the competition, that we already have the sources of all the copying, then there are two tasks in identifying plagiarism in a collection of documents: finding which pairs of documents to look at and (once a pair has been selected for further examination) finding the blocks of matching text.

In the case of the competition there *appear* to be $(7214)^2$ comparisons to be made. In the more general case usually considered by Ferret, every file needs to be compared with every other, giving $\frac{n \cdot n-1}{2}$ comparisons (which for source and suspicious files together

[*] We thank Bob Dickerson who developed the core from which the original Ferret code was developed.

is 104,076,378 pairs). But it is important to note that the way Ferret works these comparisons are not made explicitly; as the documents are read by Ferret it creates a data structure which enables the most similar pair of documents to be selected *without* making a direct comparison of those two documents. To be more specific, it remembers every three word sequence (triple) that it has seen, and which input files that triple appears in. Similar files are those with the largest number of common triples. This simple approach is what makes Ferret fast.

The Ferret user interface then allows the operator to display the similar documents side by side with the similar text highlighted. For large documents, it can take as long to display one pair as it does to find all the similarities in the set (we mention this to highlight the speed of the first phase, rather than as a deficiency of the user interface).

There is no specific support in Ferret for finding the sections of a source that has been copied. This is done by the operator (although he or she can click on any one of the matching triples to find where in the two compared documents it appears).

We expect to look at the most similar pairs (in descending order of similarity) and stop when we judge that plagiarism is no longer occurring. In effect this is a corpus specific threshold (partly mitigating the problem mentioned in section 3.2).

3 Adapting for the Competition

The problems that have to be solved in order to use Ferret for the competition relate firstly to the large volume of data to be examined and secondly to the difference between how we would normally use Ferret and how the competition is run.

3.1 Scale

To deal with the large volume of data in the competition, we had to divide the input into batches that were processed in turn as we did not have a machine with sufficient RAM to deal with all the data in one go. We estimate that 32GB would be enough; our machine was 9GB (for normal use, Ferret runs in 512KB or less). If batching is necessary, it is most efficient if half the available memory is used for source documents, and half for suspicious documents.

If M is the available memory, the number of batches of source documents, N_O, will be $\frac{2|O|}{M}$ (where $|O|$ is the total size of all the source documents in the corpus). The number of batches of suspicious documents, N_U, should be $\frac{2|U|}{M}$. The number of runs, $N_O \cdot N_U$, will thus be $\frac{4|O||U|}{M^2}$ which is quadratic in corpus size; doubling the memory available will make the system four times faster.

3.2 Automation

We needed to automate the decision between “plagiarised” and “innocent similarity”. At present Ferret supports two possible metrics: a count of the number of common triples, and the Jacquard coefficient, ‘R’.

Some of the files are very big, but these are mixed with quite small files so our current metric does not work very well. The copied chunks vary from a couple of sentences up, but it is the huge size of some of the files that causes the problem, because the number of randomly matching triples in a huge file is bigger than the size of one of the smaller copied chunks.

Examining results for the first 100 suspicious files in the Development corpus we found that we should take the 50 most similar pairs to catch all the suspicious files where artificial plagiarism had been introduced.

Ferret picks out many very small similarities. Eliminating these “accidental” similarities (common triples, typically isolated) was a problem that we had to address with code if we were to have success in the competition. It was not a problem we had seen before because of the way we use Ferret.

3.3 Improvements

Our submission was the *first* run of the complete system.

We later revised it to take some account of the order of triples in the source document when deciding if matching triples in the suspicious document are part of a matching block or just a random match. This code is quite slow, and there are now a lot of parameters that can be fiddled to change how well Ferret would perform in the competition:

- How many triples matching is considered too small to be worth considering as input to the second phase: currently less than 50 (or $R < 0.007$).

- How many documents to keep in the "most similar pairs" list: 5 on the first (submitted) run, but considering 50.
- The unmatched gap between matching sections that can be merged: 1 in the first run; considering up to 4.
- How many sections before or after the current section we can jump when merging: no restriction in first run; considering a range from 5 before to 10 after.

4 How the system operates

4.1 Identifying Similar Pairs

First we run Ferret on the complete corpus. A bash script `make-input-document-list` that creates an input file for the `ferret -f` (definition file) option. Several copies of the `make-input-document-list` script are combined in a script that does multiple runs of Ferret to do (small) groups of suspicious files against (largish) groups of source files to produce a set of output files.

4.2 Identifying Sources

We read the output files generated by step 1 to select the likely sources for a given suspicious document. This uses a ruby script `process-output` that runs `ferret -x` to produce an XML file highlighting the similarities in a particular suspicious-source pair where the number of matches and/or resemblance metric meets hard coded (but easy to change) constraints. This produces a set of XML files (one for each `ferret -x` run); these are scanned by another ruby script: `read-ferret-xml-files` to produce output in the required XML format. Formatting the results to meet the competition requirements raised a minor difficulty that the source offset required was not available in our system; fortunately it did not appear to be part of the evaluation metric.

5 Resource Analysis

Tackling more than about 500 files at a time on a 1GB laptop led to it thrashing. On a 9GB server, about 5000 files at a time could be dealt with comfortably.

For the development corpus, the output from Ferret was divided into 146 files: each file has results for about half of the source files (numbers 1-3999 or 4000-7214) and about 100 suspicious files. As explained above, this is not the best way of organising the data, but it was initially easier to test a few suspicious files at a time.

Step 1 produces a *lot* of output. As an indication of the scale of the problem, running the Unix wc (word count) utility on the complete set of output files took about 37 minutes, involving 7 minutes CPU time. The average size of the files is around 45MB, and the total size about 6.5GB.

This emphasises the quadratic nature of plagiarism detection: every suspicious file has to be checked against every possible source file. In the case of the competition this is 52,041,796 comparisons. As mentioned earlier, Ferret usually compares every file with every other, but fortunately we had already implemented a grouping facility (it has several other applications) whereby files in a group are *not* compared with each other, but only with files in other groups. We should have filtered out the least similar pairs *before* generating the output from phase 1, as the set of phase 1 output files is considerably larger than the set of suspicious documents. In normal use, Ferret displays a list of the most similar pairs; the user only looks at the most similar and because the rest of the list is in memory there is little extra cost involved.

6 Results

Here now are our observations on running the system we built around Ferret on the development corpus (we do not yet know the "correct answers" for the competition corpus).

We group the pairs into 5 types depending on the kind of artificial plagiarism:

- raw plagiarism (without obfuscation)
- low obfuscation
- high obfuscation
- plagiarism by translation
- no artificial plagiarism

We plotted the value of R for each of these types: figure 1 shows (for first 500 suspicious documents of each type in the development corpus) how documents which were identified as plagiarised (with various degrees of obfuscation) differed from those where no similarity was intended.

We see immediately that Ferret is (as expected) ineffective at detecting plagiarism by translation (the line *on top* of the x-axis), so this is left out of the later graphs.

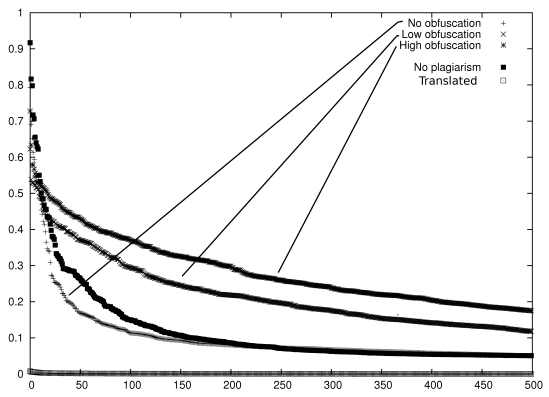

Figure 1: R for the first 500 pairs of each type

Suspicious	**Source**	R	**Cause**
00569	04692	.533	editions
01302	05069	.917	read me
01656	04163	.501	editions
01756	03972	.662	editions
01862	03766	.640	editions
02483	05054	.550	fragments
02730	01431	.629	fragments
04740	04973	.717	editions
05096	00620	.622	fragments
05959	06555	.706	editions
05964	06148	.816	editions
06188	05357	.798	plagiarism?

Table 1: Most similar non-plagiarised pairs

We also note some very high values for R, and not just where there is introduced plagiarism. Some of the pairs where there was no artificial plagiarism showed very high similarity using Ferret; there are some very high values of R before the graph flattens off. In total 49 pairs (in the development corpus) have $R > 0.5$. Twelve of these are pairs where no plagiarism is alleged. We looked at each of these pairs in detail, and present the results in Table 1.

The worst case was suspicious document 1302 which had $R = 0.91$ when compared to source document 5069. It turned out that these were both Project Gutenberg (1971-2009) "READ ME" documents with no other text. I guess this could be viewed as an accident on the part of the compilers of the corpus (Potthast, Martin *et al.* (editors), 2009). Some of the other similarities are more interesting, such as $R = 0.80$ between "The Impossibles" and "Out Like a Light", both by the same author. Despite the considerable number of small changes between the two documents, Wikipedia Authors (2009) suggest this is a re-publication of the *same* work under a new title.

Most of the high similarities in documents not alleged to be plagiarised were different editions of the same work, such as a volume which is repeated (with numerous spelling corrections) in the collected works or a "Second Edition Revised and Enlarged" with obviously a considerable overlap with the first edition.

In most of these 12 cases, we have different editions of the same work, or different collections containing most of the same material. A few are incomplete fragments, possibly arising from the way in which Project Gutenberg used to be distributed, so it is hard to tell how the similarity to other fragments came about.

We would presume that R values below 0.5 also indicate similar situations, as previous work has shown that $R > 0.04$ is the limit for "accidental similarity" but maybe a larger value is appropriate for this corpus.

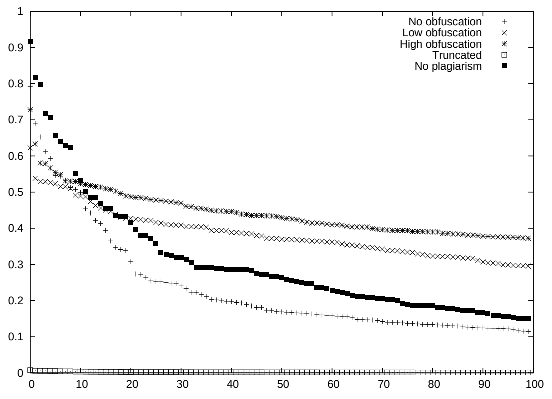

Figure 2: R for the first 100 pairs of each type

In figure 2 (where R is plotted for only the first 100 pairs of each type), it can be seen more clearly that the graphs for obfuscated plagiarism are higher than for raw, maybe because the obfuscated cases are longer: the average amount of source material in suspicious documents containing raw (un-obfuscated) plagiarism was 20,628 whereas for low and high obfuscation the average lengths were about 50% higher at 30,402 and 33,330 respectively.

Figure 3 compares R for the four types across the full range of pairs. Here the x-axis is a percentage of the total number of pairs of the type and the most similar pairs of each type ($R > 0.4$) have been omitted. The big difference between plagiarised and non plagiarised is evident, but also we note that at the RHS of the diagram these lines merge. This is because the influence on the R metric for small pieces of plagiarism in large documents is rather too small.

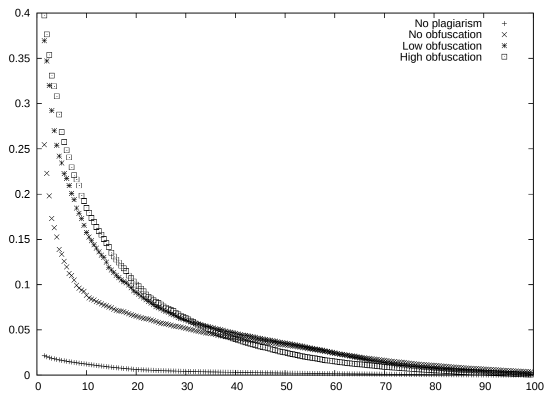

Figure 3: R for all the pairs of each type

7 Conclusions and Future Work

Producing a corpus with no similar text except that which has been added deliberately is hard: there is far too much duplicate data on the Internet to make it easy. It is for this reason that this paper concentrates on the Development corpus, as we know which parts of it are *supposed* to be plagiarised. However the competition organisers have done an excellent job in encouraging research.

The competition has clearly shown us that a better metric than R is needed. As we have suggested before (Malcolm and Lane, 2008a), a metric that takes account of the order of the similar features looks promising. Calculating the longest common subsequence of triples would probably work well, but is computationally costly; we want to take care not to slow Ferret down.

We need to develop better approaches to spanning gaps caused by obfuscation, especially in very long files as our current algorithm can still get two isolated triples that happen to be in the right order, for example a b c X k l m turning into a 7 word match, which is probably long enough to appear in the output. We should also optimise the other parameters listed in subsection 3.3.

Ferret's strength is its speed: we were able to upgrade our machine from 9 to 32GB of RAM, so can now process the entire competition corpus in a single ferret run of 1h42m. This works out at an effective rate of comparison of 50,000 pairs per second. The input is read at 450kB/s (this includes *all* ferret's processing, including writing out the very large results file).

References

Flint, Abbi, Sue Clegg, and Ranald Macdonald. 2006. Exploring staff perceptions of student plagiarism. *Journal of Further and Higher Education*, 30(2):145–156, May.

International Competition on Plagiarism Detection. 2009. `http://www.webis.de/pan-09/competition.php`.

Lyon, Caroline, Ruth Barrett, and James Malcolm. 2004. A theoretical basis to the automated detection of copying between texts, and its practical implementation in the ferret plagiarism and collusion detector. In *Plagiarism: Prevention, Practice and Policies Conference*, June.

Lyon, Caroline and James Malcolm. 2002. Experience of plagiarism detection and prevention in higher education. In *Proceedings of the World Congress, Networked Learning in a Global Environment: Challenges and Solutions for Virtual Education*. ICSC-NAISO Academic Press.

Lyon, Caroline, James Malcolm, and Ruth Barrett. 2005. The ferret copy detector: finding similar passages in large quantities of text. In *Submitted to the 43rd Annual Meeting of the Association for Computational Linguistics*.

Malcolm, J.A. and P.C.R. Lane. 2008a. An approach to detecting article spinning. *Proceedings of the Third International Conference on Plagiarism*.

Malcolm, James A. and Peter C. R. Lane. 2008b. Efficient search for plagiarism on the web. In *Proceedings of i-TCE 2008*.

Potthast, Martin *et al.* (editors). 2009. PAN Plagiarism Corpus PAN-PC-09. `http://www.webis.de/research/corpora`.

Project Gutenberg. 1971-2009. `http://www.gutenberg.org/`.

Wikipedia Authors. 2009. `http://en.wikipedia.org/wiki/Mark_Phillips_(author)`.